

title: "How to Handle Base Currency Conversion in Your App" published: true tags: api, javascript, python, tutorial canonical_url: https://allratetoday.com/blog/how-to-handle-base-currency-conversion/ cover_image:

"How do I convert from EUR to GBP if my API only supports USD as the base currency?" This question appears in nearly every exchange rate API support forum. It is one of the most common developer pain points — and it is entirely avoidable with the right API or the right code.

The Base Currency Problem

Most exchange rate APIs return rates relative to a **base (source) currency**. When you call `GET /rates?base=USD`, you get:

```
{
  "base": "USD",
  "rates": {
    "EUR": 0.9234,
    "GBP": 0.7891,
    "JPY": 151.42
  }
}
```

This tells you how much 1 USD buys in each currency. But what if your users are in Europe and you need **EUR to GBP**?

The catch: Many free API plans lock the base currency to USD. Changing the base to EUR, GBP, or any other currency requires a paid upgrade.

API	Free Base Currency	Any Base Currency
AllRatesToday	Any currency	All plans
Open Exchange Rates	USD only	Developer plan (\$12/mo)
Fixer.io	EUR only	Basic plan (\$10/mo)
CurrencyAPI	USD only	Pro plan

The Easy Way: Use an API That Supports Any Base

With AllRatesToday, just change the `source` parameter:

```
# EUR as base
curl "https://allratetoday.com/api/v1/rates?source=EUR&target=GBP" \
  -H "Authorization: Bearer YOUR_API_KEY"

# INR as base
curl "https://allratetoday.com/api/v1/rates?source=INR&target=AUD" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

This works on all plans, including the free tier.

The Manual Way: Cross-Rate Calculation

If you are stuck with a USD-locked API, you can calculate **cross-rates** yourself:

Cross-rate formula: $A/B = \text{USD_rate_B} / \text{USD_rate_A}$

Example: EUR to GBP

Given USD-based rates: - USD/EUR = 0.9234 - USD/GBP = 0.7891

Cross-rate: $\text{EUR/GBP} = 0.7891 / 0.9234 = 0.8546$

So 1 EUR = 0.8546 GBP.

JavaScript Implementation

```
function crossRate(usdRates, from, to) {
  if (from === 'USD') return usdRates[to];
  if (to === 'USD') return 1 / usdRates[from];
  return usdRates[to] / usdRates[from];
}

const usdRates = {
  EUR: 0.9234,
  GBP: 0.7891,
  JPY: 151.42,
  INR: 83.45,
  AUD: 1.5312
};
```

```
console.log('EUR to GBP:', crossRate(usdRates, 'EUR', 'GBP')); // 0.8546
console.log('GBP to JPY:', crossRate(usdRates, 'GBP', 'JPY')); // 191.89
console.log('INR to AUD:', crossRate(usdRates, 'INR', 'AUD')); // 0.01835
```

Python Implementation

```
def cross_rate(usd_rates: dict, source: str, target: str) -> float:
    """Calculate cross-rate from USD-based rates."""
    if source == "USD":
        return usd_rates[target]
    if target == "USD":
        return 1 / usd_rates[source]
    return usd_rates[target] / usd_rates[source]

usd_rates = {
    "EUR": 0.9234,
    "GBP": 0.7891,
    "JPY": 151.42,
    "INR": 83.45,
    "AUD": 1.5312,
}

print(f"EUR to GBP: {cross_rate(usd_rates, 'EUR', 'GBP'):.4f}") # 0.8546
print(f"GBP to JPY: {cross_rate(usd_rates, 'GBP', 'JPY'):.2f}") # 191.89
```

The Rounding Problem

```
// Wrong: floating-point precision issues
const amount = 1999.99;
const rate = 0.9234;
const result = amount * rate; // 1846.7907659999998

// Right: round to the target currency's decimal places
function convertAmount(amount, rate, decimals = 2) {
    return Number((amount * rate).toFixed(decimals));
}

console.log(convertAmount(1999.99, 0.9234)); // 1846.79
console.log(convertAmount(1999.99, 151.42, 0)); // 302840 (JPY has 0 decimals)
```

Currency decimal places: Most currencies use 2 decimal places, but JPY, KRW, VND use 0. BHD, KWD, OMR use 3. Always round correctly for the target currency.

Using the AllRatesToday SDK

The SDK handles base currency and conversion in a single call:

```
import AllRatesToday from '@allratetoday/sdk';

const client = new AllRatesToday('YOUR_API_KEY');

// Direct conversion – any base currency, any target
const result = await client.convert('EUR', 'GBP', 5000);
console.log(`5,000 EUR = ${result.result} GBP`);

// Get rate for any pair directly
const rate = await client.getRate('INR', 'AUD');
console.log(`1 INR = ${rate} AUD`);
```

Summary

- The base currency is the reference point for all rates in an API response.
- Many free API plans lock you to USD — forcing manual cross-rate math.
- Cross-rate formula: $A/B = \text{USD_rate_B} / \text{USD_rate_A}$
- Always round to the correct decimal places for the target currency.
- The easiest solution: use an API that supports any base currency on all plans.

AllRatesToday supports all 160+ currencies as the source — even on the free tier. No cross-rate workarounds needed.

[Get your free API key](#) | [Documentation](#) | [GitHub](#)